Loading Libraries

```
[37] %matplotlib inline
```

```
[38] from __future__ import print_function

     import torch
     import torch.nn as nn
     import torch.nn.functional as F
     import torch.optim as optim

     from PIL import Image
     import matplotlib.pyplot as plt

     import torchvision.transforms as transforms
     import torchvision.models as models

     import copy
```

Setting GPU

```
[39] device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
```

Downloading Data

```
[40] !mkdir Data
     !wget https://pytorch.org/tutorials/_static/img/neural-style/picasso.jpg
```

```
mkdir: cannot create directory ‘Data’: File exists
--2021-07-01 15:19:26--  https://pytorch.org/tutorials/_static/img/neural-style/picasso.jpg
Resolving pytorch.org (pytorch.org)... 185.199.108.153
Connecting to pytorch.org (pytorch.org)|185.199.108.153|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 114573 (112K) [image/jpeg]
Saving to: ‘picasso.jpg.2’

picasso.jpg.2       100%[===================>] 111.89K  --.-KB/s    in 0.009s

2021-07-01 15:19:26 (12.4 MB/s) - ‘picasso.jpg.2’ saved [114573/114573]
```

```
[41] !wget https://blog.upes.ac.in/wp-content/uploads/2020/08/Blog-cover-Engg.jpg
```

```
--2021-07-01 15:19:26--  https://blog.upes.ac.in/wp-content/uploads/2020/08/Blog-cover-Engg.jpg
Resolving blog.upes.ac.in (blog.upes.ac.in)... 54.163.68.27
Connecting to blog.upes.ac.in (blog.upes.ac.in)|54.163.68.27|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 166449 (163K) [image/jpeg]
Saving to: ‘Blog-cover-Engg.jpg.2’

Blog-cover-Engg.jpg 100%[===================>] 162.55K   664KB/s    in 0.2s

2021-07-01 15:19:27 (664 KB/s) - ‘Blog-cover-Engg.jpg.2’ saved [166449/166449]
```

Reshaping Image

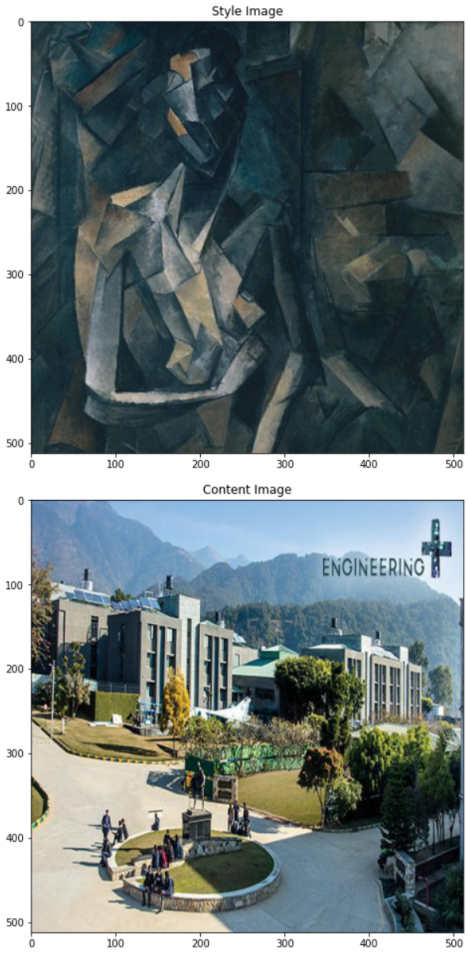
```
[42] from PIL import Image
     im=Image.open("/content/Blog-cover-Engg.jpg").resize((512,512))
     im.save("/content/Blog-cover-Engg.jpg")
```

```
[43]
     imsize = 512 if torch.cuda.is_available() else 128

     loader = transforms.Compose([
         transforms.Resize(imsize),
         transforms.ToTensor()])


     def image_loader(image_name):
         image = Image.open(image_name)

         image = loader(image).unsqueeze(0)
         return image.to(device, torch.float)


     style_img = image_loader("/content/picasso.jpg")
     content_img = image_loader("/content/Blog-cover-Engg.jpg")
     print(style_img.size())
     print(content_img.size())
```

```
torch.Size([1, 3, 512, 512])
torch.Size([1, 3, 512, 512])
```

```
[44] unloader = transforms.ToPILImage()

     plt.ion()
```

```
def imshow(tensor, title=None):
    image = tensor.cpu().clone()
    image = image.squeeze(0)
    image = unloader(image)
    plt.imshow(image)
    if title is not None:
        plt.title(title)
    plt.pause(0.001)


plt.figure(figsize=(18,8))
imshow(style_img, title='Style Image')

plt.figure(figsize=(8,8))
imshow(content_img, title='Content Image')
```



Making the Model which includes loss, gram matrix, optimizer etc

```
[45] class ContentLoss(nn.Module):

        def __init__(self, target,):
            super(ContentLoss, self).__init__()
            self.target = target.detach()

        def forward(self, input):
            self.loss = F.mse_loss(input, self.target)
            return input
```

```
[46] def gram_matrix(input):
        a, b, c, d = input.size()

        features = input.view(a * b, c * d)

        G = torch.mm(features, features.t())

        return G.div(a * b * c * d)
```

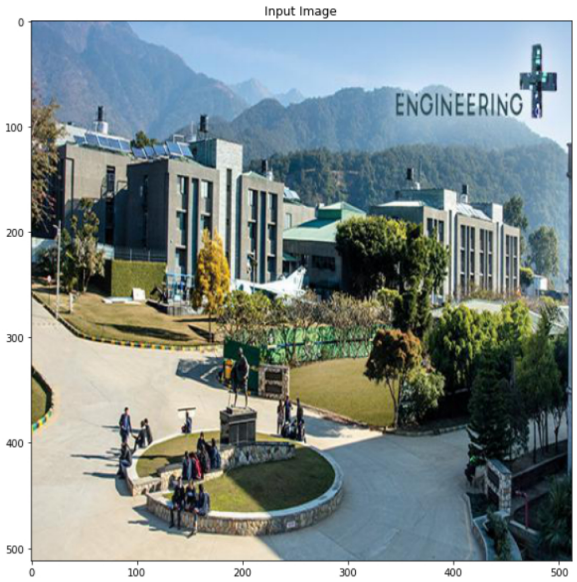
```
[47] class StyleLoss(nn.Module):

        def __init__(self, target_feature):
            super(StyleLoss, self).__init__()
            self.target = gram_matrix(target_feature).detach()

        def forward(self, input):
            G = gram_matrix(input)
            self.loss = F.mse_loss(G, self.target)
            return input
```

[48]
```
cnn = models.vgg19(pretrained=True).features.to(device).eval()
```

[49]
```
cnn_normalization_mean = torch.tensor([0.485, 0.456, 0.406]).to(device)
cnn_normalization_std = torch.tensor([0.229, 0.224, 0.225]).to(device)

class Normalization(nn.Module):
    def __init__(self, mean, std):
        super(Normalization, self).__init__()
        self.mean = torch.tensor(mean).view(-1, 1, 1)
        self.std = torch.tensor(std).view(-1, 1, 1)

    def forward(self, img):
        return (img - self.mean) / self.std
```

[50]
```
content_layers_default = ['conv_4']
style_layers_default = ['conv_1', 'conv_2', 'conv_3', 'conv_4', 'conv_5']

def get_style_model_and_losses(cnn, normalization_mean, normalization_std,
                               style_img, content_img,
                               content_layers=content_layers_default,
                               style_layers=style_layers_default):
    cnn = copy.deepcopy(cnn)
    normalization = Normalization(normalization_mean, normalization_std).to(device)

    content_losses = []
    style_losses = []

    model = nn.Sequential(normalization)

    i = 0
    for layer in cnn.children():
        if isinstance(layer, nn.Conv2d):
            i += 1
            name = 'conv_{}'.format(i)
        elif isinstance(layer, nn.ReLU):
            name = 'relu_{}'.format(i)
            layer = nn.ReLU(inplace=False)
        elif isinstance(layer, nn.MaxPool2d):
            name = 'pool_{}'.format(i)
        elif isinstance(layer, nn.BatchNorm2d):
            name = 'bn_{}'.format(i)
        else:
            raise RuntimeError('Unrecognized layer: {}'.format(layer.__class__.__name__))

        model.add_module(name, layer)

        if name in content_layers:
            target = model(content_img).detach()
            content_loss = ContentLoss(target)
            model.add_module("content_loss_{}".format(i), content_loss)
            content_losses.append(content_loss)

        if name in style_layers:
            target_feature = model(style_img).detach()
            style_loss = StyleLoss(target_feature)
            model.add_module("style_loss_{}".format(i), style_loss)
            style_losses.append(style_loss)

    for i in range(len(model) - 1, -1, -1 ):
        if isinstance(model[i], ContentLoss) or isinstance(model[i], StyleLoss):
            break

    model = model[:(i + 1)]

    return model, style_losses, content_losses
```

[51]
```
input_img = content_img.clone()
plt.figure(figsize=(10,10))
imshow(input_img, title='Input Image')
```



[52]
```
def get_input_optimizer(input_img):
```

```
        optimizer = optim.LBFGS([input_img.requires_grad_()])
        return optimizer
```

```
[53] def run_style_transfer(cnn, normalization_mean, normalization_std,
                            content_img, style_img, input_img, num_steps=300,
                            style_weight=1000000, content_weight=1):
        """Run the style transfer."""
        print('Building the style transfer model..')
        model, style_losses, content_losses = get_style_model_and_losses(cnn,
            normalization_mean, normalization_std, style_img, content_img)
        optimizer = get_input_optimizer(input_img)

        print('Optimizing..')
        run = [0]
        while run[0] <= num_steps:

            def closure():
                input_img.data.clamp_(0, 1)

                optimizer.zero_grad()
                model(input_img)
                style_score = 0
                content_score = 0

                for sl in style_losses:
                    style_score += sl.loss
                for cl in content_losses:
                    content_score += cl.loss

                style_score *= style_weight
                content_score *= content_weight

                loss = style_score + content_score
                loss.backward()

                run[0] += 1
                if run[0] % 50 == 0:
                    print("run {}:".format(run))
                    print('Style Loss : {:4f} Content Loss: {:4f}'.format(
                        style_score.item(), content_score.item()))
                    print()

                return style_score + content_score

            optimizer.step(closure)

        input_img.data.clamp_(0, 1)

        return input_img
```

Finally, we can run the algorithm.

```
output = run_style_transfer(cnn, cnn_normalization_mean, cnn_normalization_std,
                            content_img, style_img, input_img)

plt.figure(figsize=(12,12))
imshow(output, title='Output Image')


plt.ioff()
plt.show()
```

```
Style Loss : 67.262604 Content Loss: 18.035416

run [100]:
Style Loss : 35.102474 Content Loss: 14.667629

run [150]:
Style Loss : 20.117565 Content Loss: 12.357466

run [200]:
Style Loss : 11.402240 Content Loss: 11.185009

run [250]:
Style Loss : 6.167054 Content Loss: 10.537485

run [300]:
Style Loss : 3.017264 Content Loss: 10.174667
```